# PHYS 643: Writing Hydro Codes

Eve J. Lee

Notes adapted from Prof. Andrew Cumming (McGill) and Dr. Paul Duffell (Harvard CfA).

## 1 Introduction

In general, fluid equations cannot be solved analytically, and instead need to be solved numerically. Here, we will cover the *basics* of how to write your own hydro solver. I'm only giving a very brief outline here. For more detailed discussions, you may consult the Heidelberg lectures (`http://www.ita.uni-heidelberg.de/~dullemond/lectures/num_fluid_2011/`)

## 2 First Order Finite Differencing

The whole idea behind this method is to define some fluid quantity as a function of time and space over some discrete spatial grid. Consider 1-dimensional advection-diffusion equation of some variable $f$:

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = D\frac{\partial^2 f}{\partial x^2} \tag{1}$$

where $D$ is some constant diffusivity and velocity $u$ is assumed constant. This should look a lot like Navier-Stokes equation for incompressible flows in the absence of gravity and pressure gradient forces.

We will calculate the quantity $f$ on a 1-dimensional grid such that $f_j$ is the value of $f$ at $x = x_j$ where $j = 1$ to $N$ ($N$ is an integer). For simplicity, we will consider uniform grid spacing: $x_{j+1} - x_j = \Delta_x$.[1] In order to solve the advection-diffusion equation, we need to define the derivative of $f$, $f'_j \equiv \partial_x f$ at $x = x_j$, and $f''_j \equiv \partial_x^2 f$ at $x = x_j$. From Taylor expansion:

$$f_{j+1} = f_j + \Delta_x f'_j + \frac{\Delta_x^2}{2} f''_j + O(\Delta_x^3) \tag{2}$$

$$f_{j-1} = f_j - \Delta_x f'_j + \frac{\Delta_x^2}{2} f''_j - O(\Delta_x^3) \tag{3}$$

We can subtract these two to obtain an expression for $f'_j$:

$$f'_j = \frac{f_{j+1} - f_{j-1}}{2\Delta_x} + O(\Delta_x^2). \tag{4}$$

Notice how the error is on the order of $\Delta_x^2$; this expression is accurate to second order.

By adding equations 2 and 3, we can define $f''_j$:

$$f''_j = \frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta_x^2} + O(\Delta_x^2). \tag{5}$$

We have all the required expressions to define the time evolution of the quantity $f$! Let us first consider just the advection:

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0 \tag{6}$$

We will label $f$ at different time with superscript $n$ so that the above advection equation can be written as:

$$\frac{f_j^{n+1} - f_j^n}{\Delta_t} = -u\frac{f_{j+1}^n - f_{j-1}^n}{2\Delta_x}, \tag{7}$$

[1] In real-life research application, we often use variable grid spacing that is based on the resolution requirement ("adaptive mesh refinement") or based on some sort of tessellation scheme.

so that I can update the value of $f_j$ in time according to

$$f_j^{n+1} = f_j^n - \frac{u\Delta_t}{2\Delta_x}(f_{j+1}^n - f_{j-1}^n). \tag{8}$$

Since $f^{n+1}$ is defined only in terms of $f^n$, this is called "explicit" method. In fact, this particular definition is called Forward-Time Central-Space (FTCS) method. If you try to solve the advection equation using the FTCS method, you will find that your numerical solution is unstable!

## 2.1 Von Neumann Stability Analysis

How can we assess a priori whether the chosen numerical method will be stable vs. unstable? In general, a solution can be expressed as a finite Fourier series with amplitude $\xi^n$ that depends on time. Let's consider just one component:

$$f_j^n = \xi^n e^{ik(j\Delta_x)}. \tag{9}$$

For the solution to be stable, we need to ensure $|\xi^{n+1}/\xi^n| < 1$ for all wavenumber $k$; otherwise, $f$ will grow exponentially with time. Let's substitute equation 9 into equation 8 to assess the stability of FTCS scheme:

$$\begin{aligned}
e^{ik(j\Delta_x)}\xi^{n+1} &= e^{ik(j\Delta_x)}\xi^n - \frac{u\Delta_t}{2\Delta_x}(\xi^n e^{ik(j+1)\Delta_x} - \xi^n e^{ik(j-1)\Delta_x}) \\
\frac{\xi^{n+1}}{\xi^n} &= 1 - \frac{u\Delta_t}{2\Delta_x}(e^{ik\Delta_x} - e^{-ik\Delta_x}) \\
&= 1 - \frac{u\Delta_t}{\Delta_x} i \sin k\Delta_x \\
\left|\frac{\xi^{n+1}}{\xi^n}\right| &= \left|1 + \left(\frac{u\Delta_t}{\Delta_x}\right)^2 \sin^2(k\Delta_x)\right|,
\end{aligned} \tag{10}$$

which is always larger than 1 so FTCS scheme is unstable.

## 2.2 Lax-Friedrichs method

We replace equation 8 with

$$f_j^{n+1} = \frac{1}{2}(f_{j+1}^n + f_{j-1}^n) - \frac{u\Delta_t}{2\Delta_x}(f_{j+1}^n - f_{j-1}^n). \tag{11}$$

From stability analysis,

$$\begin{aligned}
\left|\frac{\xi^{n+1}}{\xi^n}\right| &= \left|\cos^2(k\Delta_x) + \left(\frac{u\Delta_t}{\Delta_x}\right)^2 \sin^2(k\Delta_x)\right| \\
&= \left|1 + \sin^2(k\Delta_x)\left[\left(\frac{u\Delta_t}{\Delta_x}\right)^2 - 1\right]\right|.
\end{aligned} \tag{12}$$

The Lax-Friedrichs method is stable if

$$\Delta_t \leq \Delta_x/u, \tag{13}$$

i.e., the timestep of integration needs to be shorter than the advection time across a grid cell. This is called the "Courant" condition.

Note that computational error emerges as "numerical viscosity". Equation 11 can be re-written as

$$\frac{f_j^{n+1} - f_j^n}{\Delta_t} = -u\left(\frac{f_{j+1}^n - f_{j-1}^n}{2\Delta_x}\right) + \frac{1}{2}\left(\frac{f_{j+1}^n - 2f_j^n + f_{j-1}^n}{\Delta_t}\right), \tag{14}$$

which is the FTCS representation of

$$\frac{\partial f}{\partial t} = -u\frac{\partial f}{\partial x} + \frac{\Delta_x^2}{2\Delta_t}\frac{\partial^2 f}{\partial x^2} \tag{15}$$

where the last term represents numerical viscosity.

## 2.3 Godunov method (Upwind differencing)

Note that in Lax-Friedrichs method, the information of both the $(j+1)$'th and the $j$'th cells propagate to the $j$'th cell which is unphysical given that the advection velocity $u$ has one direction. To avoid this, we further make the following adjustment:

$$f_j^{n+1} = \begin{cases} f_j^n - u_j^n \frac{\Delta_t}{\Delta_x}(f_j^n - f_{j-1}^n) & \text{if } u_j^n > 0 \\ f_j^n - u_j^n \frac{\Delta_t}{\Delta_x}(f_{j+1}^n - f_j^n) & \text{if } u_j^n < 0 \end{cases} \tag{16}$$

where $u_j^n$ is the advection velocity at grid $j$ at time $n$. The same Courant condition as before needs to be satisfied for stability.

## 2.4 Taking Care of Diffusion

Isolating the diffusion term:

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}. \tag{17}$$

We could use finite differencing but the computation time scales quadratically with the grid size. Instead, use an implicit scheme:

$$\frac{f_j^{n+1} - f_j^n}{\Delta_t} = \frac{D}{(\Delta_x)^2}(f_{j+1}^{n+1} - 2f_j^{n+1} + f_{j-1}^{n+1}) \tag{18}$$

which can be re-written as

$$-\beta f_{j+1}^{n+1} + (1+2\beta)f_j^{n+1} - \beta f_{j-1}^{n+1} = f_j^n \tag{19}$$

with $\beta = D\Delta t/(\Delta x)^2$. This can be further re-written as $A\vec{f}^{n+1} = \vec{f}^n$ with $A$ a tri-diagonal matrix with elements -$\beta$, $(1+2\beta)$, and -$\beta$ so that $\vec{f}^{n+1}$ can be solved by inverting $A$: $\vec{f}^{n+1} = A^{-1}\vec{f}^n$.

## 2.5 Operator Splitting

Now that we know how to solve advection and diffusion equations separately, how do we solve the advection-diffusion equation? Simply add them by means of splitting the operator.

### 2.5.1 Explicit method

First update your grid with the diffusion term:

$$f_j^{n+1} = f_j^n + \beta(f_{j+1}^{n+1} - 2f_j^{n+1} + f_{j-1}^{n+1}), \tag{20}$$

then calculate the advection term:

$$f_j^{n+1} = \frac{1}{2}(f_{j+1}^n + f_{j-1}^n) - \alpha(f_{j+1}^n - f_{j-1}^n) \tag{21}$$

with $\alpha = u\Delta_t/\Delta_x$.

Here's a Python code snippet with T some array with n elements:

```
T[1:n-1] += beta * (T[:n-2] + T[2:] - 2.0 * T[1:n-1]) #diffusion
T[1:n-1] = 0.5 * (T[:n-2] + T[2:]) - 0.5 * alpha * (-T[:n-2] + T[2:]) #advection
```

### 2.5.2 Implicit method

As before, first update your grid with the diffusion term but using the implicit scheme. Here's a Python code snippet on calculating the diffusion term:

```
import numpy as np

A = np.eye(n) * (1.0 + 2.0 * beta) + np.eye(n, k=1) * -beta + np.eye(n, k=-1)*-beta
#Apply appropriate boundary conditions on A
T = np.linalg.solve(A, T)
```

Advection terms can be calculated afterward as before.

### 2.5.3 Boundary conditions on the matrix A

There are two different boundary conditions on the fluid velocity when it comes to diffusive motion: no-slip (zero relative velocity with respect to the hard boundary) and stress-free (zero velocity gradient). Let's consider the hard surface to be situated at the left boundary of the grid and impose stress-free condition at the right boundary of the grid. You should have the initial condition on the fluid velocity such that v[0] is set to the speed of the boundary. Then, you need to ensure v[0] stays fixed at all times:

```
A[0][0] = 1
A[0][1] = 0
```

This ensures the first element is fixed and that there is no diffusive flux through the first element.

What about the stress-free condition at the right boundary? You want to make sure that the $f_{j+1} - f_j$ term and the $f_j - f_{j-1}$ term goes to zero at the right-most boundary in equation 19. This can be done by

```
A[-1][-1] = 1 + beta
```

## 3 Finite Volume Methods

Recall that fluid equations are nothing more than statements of conservation (conservation of mass, conservation of momentum, conservation of energy), and follow the form

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}(uf) = (\text{sources and sinks}) \tag{22}$$

(i.e., the rate of change of $f$ + the flux of $f$ is controlled by the sources and sinks of $f$).

In this method, we divide the volume into cells and we track the rate of change of $f$ in the j-th cell by calculating the net flux of $f$ through the cell:

$$\frac{f_j^{n+1} - f_j^n}{\Delta_t} = \frac{J_{j-1/2} - J_{j+1/2}}{\Delta_x} \tag{23}$$

where the cell location is $x_j$ and its boundaries are $x_{j-1/2}$ and $x_{j+1/2}$. We need to define the flux $J$.

## 3.1 Donor Cell Advection: Hydro code example

Letting $f_1 = \rho$ and $f_2 = \rho u$, with $P = \rho c_s^2 = f_1 c_s^2$ and $c_s$ being constant, the continuity and the Euler equation (in the absence of body force) can be written as

$$\begin{aligned} &\frac{\partial f_1}{\partial t} + \frac{\partial}{\partial x}(uf_1) = 0 \\ &\frac{\partial f_2}{\partial t} + \frac{\partial}{\partial x}(uf_2) = -\frac{\partial P}{\partial x}. \end{aligned} \tag{24}$$

### 3.1.1 First consider no source term

From continuity equation, we can update

$$f_{1,j}^{n+1/2} = f_{1,j}^n - \frac{\Delta_t}{\Delta_x}(J_{1,j+1/2} - J_{1,j-1/2}) \tag{25}$$

In donor cell advection scheme, we assume $f_1$ and $f_2$ are constant within the cell so that

$$J_{1,j+1/2} = \begin{cases} u_{j+1/2}^n f_{1,j}^n, & u_{j+1/2}^n > 0 \\ u_{j+1/2}^n f_{1,j+1}^n, & u_{j+1/2} < 0 \end{cases}$$

and

$$J_{1,j-1/2} = \begin{cases} u_{j-1/2}^n f_{1,j-1}^n, & u_{j-1/2} > 0 \\ u_{j-1/2}^n f_{1,j}^n, & u_{j-1/2} < 0 \end{cases}$$

where the velocity at the cell interface is defined as

$$u_{j+1/2}^n = \frac{1}{2}\left(\frac{f_{2,j}}{f_{1,j}} + \frac{f_{2,j+1}}{f_{1,j+1}}\right). \tag{26}$$

We follow the same scheme for $f_2$ with $\partial P/\partial x = 0$.

### 3.1.2 Add the source term

No explicit source term for continuity so mass is simply conserved:

$$f_{1,j}^{n+1} = f_{1,j}^{n+1/2}. \tag{27}$$

For the Euler equation, we need to consider the pressure gradient:

$$f_{2,j}^{n+1} = f_{2,j}^{n+1/2} - \frac{\Delta_t}{\Delta_x} c_s^2 (f_{1,j+1}^{n+1/2} - f_{1,j-1}^{n+1/2}) \tag{28}$$

### 3.1.3 Boundary conditions

There are many different kinds of boundary conditions. The most common ones are: fixed boundaries, outflow (flows past the simulation boundary are lost forever), periodic (i.e., loops back to other side), or reflective. We'll only consider reflective boundaries here. Simply apply:

```
f1[0] = f1[0] - (dt / dx) * J1[0]
f1[-1] = f1[-1] + (dt / dx) * J1[-2]
```

and likewise for $f_2$ and $J_2$ to "reflect" the advection term. For the computational implementation for donor cell advection, refer to `http://www.ita.uni-heidelberg.de/~dullemond/lectures/num_fluid_2011/Chapter_5.pdf`.